IN THE CLAIMS:

Please amend claims 1, 2, 5 and 6 as follows:

Claim 1. (currently amended) A method for operating an out-of-order processor comprised of an instruction pipeline, the method comprising the steps of:

for detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions is not the same as a logic source address of said current instruction, said one or more instructions being stored in a temporary buffer associated with a pipeline process downstream of the current instruction;

generating a no-dependency signal associated with said current instruction; and

~~if the no-dependency signal is not active, assigning an entry in the temporary buffer to the logic source address of said current instruction; and~~

~~if the no-dependency signal is active, issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction~~

bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active.

Claim 2. (currently amended) The method according to claim 1 in which the step of generating a ~~no-dependency~~ no-dependency signal comprises the steps of:

comparing a plurality of logic target register addresses and the logic source register address of the current instruction;

in case the logic target register addresses and the logic source register address match, setting the no-dependency signal ~~is~~ to not active; and

generating a dependency signal for the respective source register.

Claim 3. (previously presented) The method according to claim 1 further comprising the step of evaluating 'valid'-bits of speculative target registers stored in a storage associated with speculatively calculated instruction result data to generate the no-dependency signal.

Claim 4. (previously presented) The method according to claim 1 further comprising the step of applying the method to a mapping table based renaming scheme comprising the steps of:

addressing a mapping-table-entry with a logical source register address of said current instruction thus determining the mapped physical target register address;

reading a committed-status flag in said entry;

comparing the logic target register address and the logic source register address of the current instruction in case the no-dependency signal is not active; and

generating a dependency signal for the respective source register.

Claim 5. (currently amended) The method according to claim ~~2~~ 1 further comprising the step of applying the method to a mapping-table-based renaming scheme comprising the steps of:

addressing a mapping table entry with a logical source register address of said current instruction thus determining the mapped physical target register address;

reading a committed-status flag in said entry;

comparing the logic target register address and the logic source register address of the current instruction;

in case the logic target register address and the logic source register address match, setting the no-dependency signal ~~is~~ to not active; and

generating a dependency-signal for the respective source register.

Claim 6. (currently amended) A processing system having means for executing a readable machine language, said readable machine language comprises:

a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction,

a second computer readable code for generating a no-dependency signal associated with said current instruction, and

a third computer readable code for bypassing a portion of the instruction pipeline for the current instruction if the no-dependency signal is active ~~assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active; and~~

~~a fourth computer readable code for issuing the instruction operand data to an instruction execution unit without assigning the entry in the~~

~~temporary buffer to the logic source address of said current instruction if the no-dependency signal is active~~.

Claim 7. (previously presented) The processing system according to claim 6 in which in case of a content-addressable memory (CAM)-based renaming scheme the first computer readable code for determining the dependency of a current instruction comprises a compare logic in which all instructions to be checked for dependency are involved and an OR gate coupled with the compare logic.

Claim 8. (previously presented) The processing system according to claim 7 further comprising a plurality of AND gates the input of which comprises a target register 'valid bits' signal and a respective compare logic output signal.

Claim 9. (previously presented) The processing system according to claim 6 in which the case of a mapping-table-based renaming scheme each mapping table entry comprises an additional instruction-commited flag, and the first computer readable code for determining the dependency of a current instruction comprises a logic for ANDing a target register 'valid bits' signal in which all instructions to be checked for dependency are involved and an OR gate coupled with the logic.

Claim 10. (previously presented) A computer system having an out-of-order processing system, said computer system executes a readable machine language, said readable machine language comprises:

a first computer readable code for, the detection of a dependency, determining for each current instruction involved in a renaming process that a logic target address of one or more instructions stored in a temporary buffer associated with a pipeline process downstream of the current instruction is not the same as a logic source address of said current instruction,

a second computer readable code for generating a no-dependency signal associated with said current instruction,

a third computer readable code for assigning an entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is not active; and

a fourth computer readable code for issuing the instruction operand data to an instruction execution unit without assigning the entry in the temporary buffer to the logic source address of said current instruction if the no-dependency signal is active.